Custom URLs using the missing File handler

Posted At : April 29, 2005 3:42 PM | Posted By : Mark Kruger
Related Categories: Coldfusion Tips and Techniques

(reprinted from a previous blog)

If you have ever worked on a community site, or a site with salesman profiles or profiles of individuals, you may have said to yourself, "Self, it sure would be nice if each of these folks could have their own website". Doing it would require you to make a unique host entry for each user or create separate folders for each user and of course, that's not really easily managed. Or is it? Perhaps there's another way. Actually IIS and Cold Fusion give you a way to accomplish this pretty easily. 2 ways in fact.

> The first method is through the use of the dreaded "missing file template". When IIS can't find a file to serve it has 2 choices. 1 - it can send a "missing file status code" back to the browser in the response header. The actual code is 404, which is why "404 error" has become a euphemism for "missing file". The second choice is to actually serve a file. What file you ask? Why.... whatever file is specified in the "custom errors" for that site. By default, in fact, IIS comes equipped with a host of custom error templates. These are simple HTML files that are served in leu of the actual status code. They say extremely informative things like "404 error - file not found". Honestly, you'd think as much cash as Microsoft has they would hire someone other than Dilbert's boss to write their error messages.

Ok, here's the deal. IIS error messages need *not* be written by Microsoft. In fact you can write them yourself or specify any file you like to be served. You can hire bolivian contractors to write them, warn people about global warning instead of throwing an error, or even (dare I say it) display something actually helpful - like a link to your support page for further help. In fact (and here's where our solution comes in), you don't actually have to use a "file" at all - you can use a URL from your site. That's work CF comes in.

Try this:

1. Open your site properties and choose "custom errors"
2. Click on the 404 custom error and choose "edit"
3. Change the message type to "URL"
4. Enter the relative path to a .cfm script on your site.

Remember, you *don't* enter a full URL here - the file *must* reside on your site and be accessible from a relative path. Why? Well, IIS is going to process the this value as if it were the file part of an HTTP request. Remember what an HTTP header looks like?

```
GET /mycode/myPage.cfm      HTTP/1.1
   Content-Type: text/html
   Referer:
   User-Agent: Mozilla/4.0
   Host:   blog.mxconsulting.com
   Content-Length: 488
   .....
```

The Host address is separate from the "GET" part of the request. If you put in a full URL IIS would actually have to retrieve content using HTTP using a completely separate request. Instead, IIS wants to simply replace the erroneous part of the header information with the "known good" file mapping and continue on with the request as if it had been for the missing file template in the first place. That means that a request that hits the missing file template is about as fast as if the file specified had existed in the first place.

So, if IIS merely replaces the non-existent "/mycode/myPage.cfm" with a mapping from the custom error handler setting from the 404 error and continues on, does that mean I can use a .cfm page for my 404 handler? Yes indeed - it means exactly that! If you have specified "404Handler.cfm" and set the message type to URL in the custom error page. The request continues as before and the it is handed off to the Coldfusion server for processing with the original path sent forward as a the cgi.query_string variable. CF server does its thing, parsing through the script looking for commands and cf tags and sends the results back to IIS which serves it up to the user.

That being the case it's pretty easy to imagine how to use this to your advantage if you wanted to create a custom URL for a community site, or custom directories for a real estate site (to use another example). In the file handler simply parse out the string that represents the missing directory and use it as a variable to determine what actually *should* be served up to the user. For example, if you wanted to use the zip code for house searches on a real estate site you could have the user enter "www.myhomes.com/68154". Since there *is no* directory titled 68154, this request would be forwarded to the missing file handler - we'll call it "haveYouSeenThisFile.cfm" with a querystring that contains ";404/68156" as the last item. All you have to do is `<zipcode = Listlast(cgi.query_string,'/')>"` and you have the zip code being sought. Display or redirect the user to the page with houses from that zip code and you are done.

DNS wildcard

The second method is actually cooler - but it can confuse people. First, set a wild card entry in your DNS server that points to your web server (or have your DNS provider do it. This entry is a rule that says, *"If you receive a request for DNS resolution and there is no specific match for that string - forward it to the web server."* In other words, if your DNS has an entry for "mail.mydomain.com" and a user requests that IP address then that specific IP address is given in response. If however a request is received for Bill.mydomain.com and there *is no entry for Bill* (why would there be) then that request is forwarded to the web server.

To use our zip code example, a user could enter "68156.myhomes.com" in their browser and that request would be forwarded to the web server. On the web server, the default document (a .cfm file - let's say "index.cfm") examines the value of cgi.Http_Host (usually www.myhomes.com) looking at Listfirst(cgi.Http_host,"."). If that item is not "www" and not "myhomes" then it takes it to be a zip code and directs the user accordingly - simple right?

Well actually this approach works well if you are sending out links to people via email or you have links on a page or web site or in a document. But if you have a need to actually speak or explain the URL to someone it works less well. Most novice users simply don't grasp that a web site doesn't need a "www" in the URL. So you end up with www.68283.myhomes.com or other variations. You can account for this in your parsing code of course, but you have added complexity to the process instead of making it easier (which *is* the point of this whole thing after all - right?).

Either approach can result in a nicely customized URL for many uses and a very minimal code base accounting for it. Neither requires extra "hard to track and see" things like mod_rewrite or ISAPI filters. And neither approach incurs a heavy performance penalty.